import { reactive } from 'vue'

浅谈 Vue 3 响应式与组合式 API 的一些应用

Anthony Fu / @antfu / 2020

Anthony Fu

Core Team	Vue.js, Vite.js
Focus	@vue/composition-api, vite
GitHub	/antfu
Twitter	@antfu7
Blog	//antfu.me

//antfu.me/talks/2020-09-26

Overview

- 响应式 API & 组合式 API
- 组合式的一个例子
- Vue 2 和 3 工具库组合式同构
- 响应式延伸应用

3.0 Released 🎉

One Piece

Vue 3.0

```
                                 +---------------------+
                                 |                     |
                                 |  @vue/compiler-sfc  |
                                 |                     |
                                 +-----+--------+------+
                                       |        |
                                       v        v
                   +---------------------+    +----------------------+
                   |                     |    |                      |
     +------------>|  @vue/compiler-dom  +--->|  @vue/compiler-core  |
     |             |                     |    |                      |
+----+----+        +---------------------+    +----------------------+
|         |
|   vue   |
|         |
+----+----+        +---------------------+    +----------------------+    +-------------------+
     |             |                     |    |                      |    |                   |
     +------------>|  @vue/runtime-dom   +--->|  @vue/runtime-core   +--->|  @vue/reactivity  |
                   |                     |    |                      |    |                   |
                   +---------------------+    +----------------------+    +-------------------+
```

Vue 3.0

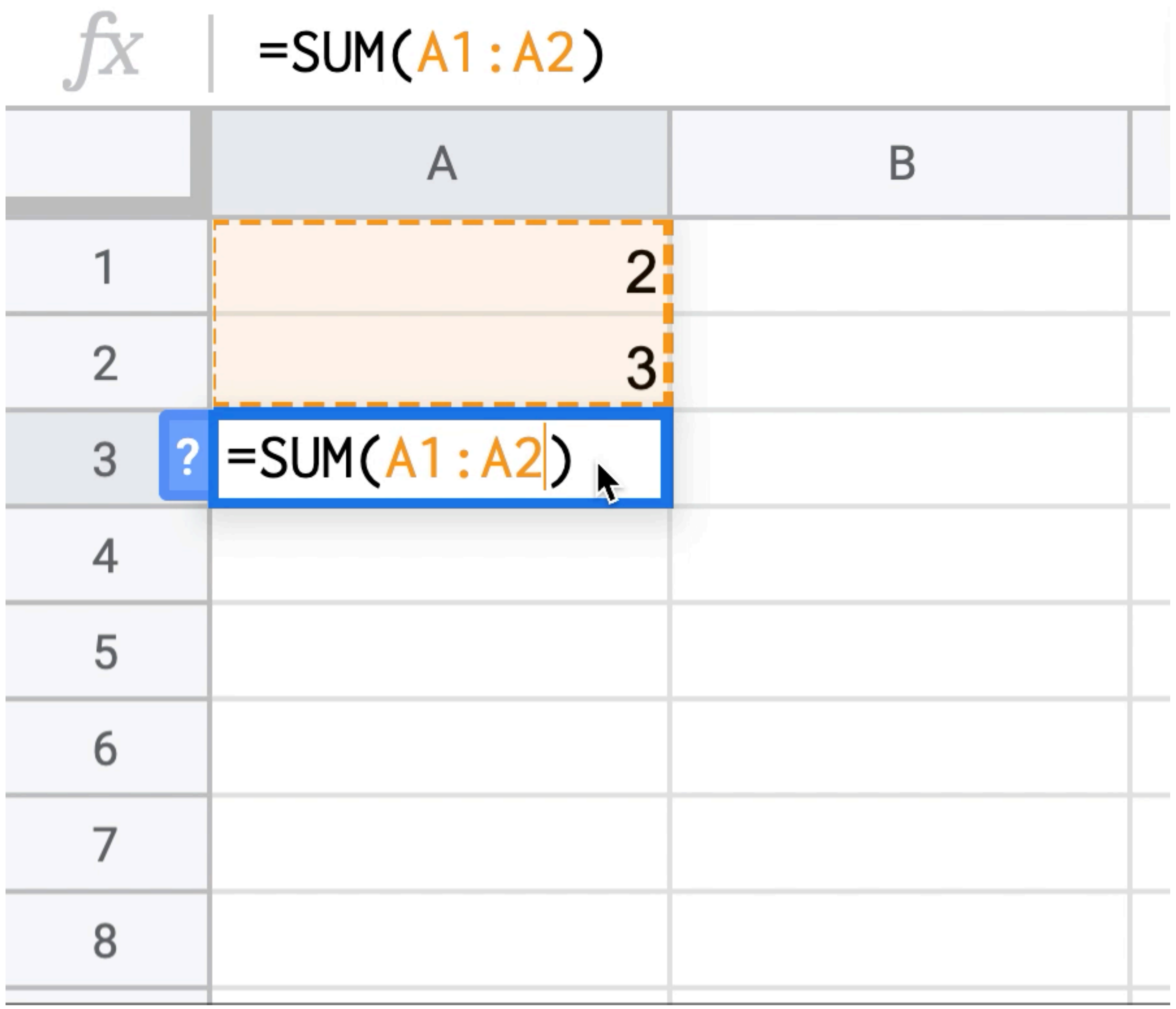
Composition & Reactivity

```
                                    +---------------------+
                                    |                     |
                                    |  @vue/compiler-sfc  |
                                    |                     |
                                    +-----+--------+------+
                                          |        |
                                          v        v
                      +---------------------+    +----------------------+
                      |                     |    |                      |
        +------------>|  @vue/compiler-dom  +--->|  @vue/compiler-core  |
        |             |                     |    |                      |
   +----+----+        +---------------------+    +----------------------+
   |         |
   |   vue   |
   |         |                                          "组合式"                   "响应式"
   +----+----+        +---------------------+    +----------------------+    +-------------------+
        |             |                     |    |                      |    |                   |
        +------------>|  @vue/runtime-dom   +--->|  @vue/runtime-core   +--->|  @vue/reactivity  |
                      |                     |    |                      |    |                   |
                      +---------------------+    +----------------------+    +-------------------+
```

响应式

Reactivity

- 自动收集依赖&更新
- Vue 3 新 API
  - `ref`
  - `reactive`
  - `effect`
  - `computed`



响应式 - Reactive

Reactivity

- 使用 Proxy 实现
- track, trigger 进行响应式追踪

```
const reactive = (target) => new Proxy(target, {
  get(target, prop, receiver) {
    track(target, prop)
    return Reflect.get(...arguments) // get original data
  },
  set(target, key, value, receiver) {
    trigger(target, key)
    return Reflect.set(...arguments)
  }
})

const obj = reactive({
  hello: 'world'
})

console.log(obj.hello) // `track()` get called
obj.hello = 'vue' // `trigger()` get called
```

响应式 - Effect

Reactivity

- track - 追踪调用它的函数
- trigger - 触发绑定的更新
- effect - 调用函数并收集依赖

```
const targetMap = new WeakMap()

export const track = (target, key) => {
  if (tacking && activeEffect)
    targetMap.get(target).key(key).push(activeEffect)
}

export const trigger = (target, key) => {
  targetMap.get(target).key(key).forEach(effect => effect())
}

export const effect = (fn) => {
  let effect = function() { fn() }
  enableTracking()
  activeEffect = effect
  fn()
  resetTracking()
  activeEffect = undefined
}
```

响应式 - Computed

Reactivity

- Computed 和 Watch
  - 基于 Effect 的包装
- 延伸阅读

```
const computed = (getter) => {
  let value
  let dirty = true

  const runner = effect(getter, {
    lazy: true,
    scheduler() {
      dirty = true // deps changed
    }
  })

  return {
    get value() {
      if (dirty) {
        value = runner() // re-evaluate
        dirty = false
      }
      return value
    }
  }
}
```

组合式
Composition

- 基于响应式
- 提供 Vue 的生命周期钩子 (onMounted, onUnmounted, etc.)
- 组件销毁时自动销毁依赖监听 (computed, watch, etc.)
- 可复用的逻辑
    - 跨组件复用
    - 跨应用复用 (Composable Libraries)

响应式

@vue/reactivity

`ref`

`reactive`

`computed`

`effect`

组合式

@vue/runtime-core

`watch`

`setup`

`onMounted (lifecycles)`

`getCurrentInstance`

响应式

@vue/reactivity

ref

reactive

computed

effect

可以作为一个独立的包使用

组合式

@vue/runtime-core

watch

setup

onMounted (lifecycles)

getCurrentInstance

组合式 API 使用场景

Case Study

Dark mode

一个例子

- 默认跟随系统偏好
- 可手动更改模式
- 可持久化
- 模式改变时执行一些操作

Dark mode

一个例子

- 默认跟随系统偏好
- 可手动更改模式
- 可持久化
- 模式改变时执行一些操作

基础功能

```
<template>
  <div :class='{dark}'>
    <button @click='toggleDark'>Toggle</button>
  </div>
</template>
```

基础功能

Options API

```
<script>
export default {
  data() {
    return {
      dark: false
    }
  },
  methods: {
    toggleDark() {
      this.dark = !this.dark
    }
  }
}
</script>
```

Composition API

```
<script>
import { ref } from 'vue'

export default {
  setup() {
    const dark = ref(false)

    return {
      dark,
      toggleDark() {
        dark.value = !dark.value
      }
    }
  }
}
</script>
```

增加 Media Query

Options API

```
<script>
export default {
  data() {
    return {
      dark: false,
      media: window.matchMedia('(prefers-color-scheme: dark)')
    }
  },
  methods: {
    toggleDark() {
      this.dark = !this.dark
    },
    update() {
      this.dark = this.media.matches
    }
  },
  created() {
    this.media.addEventListener('change', this.update)
    this.update()
  },
  destroyed() {
    this.media.removeEventListener('change', this.update)
  }
}
</script>
```

Composition API

```
<script>
import { onUnmounted, ref } from 'vue'

export default {
  setup() {
    const media = window.matchMedia('(prefers-color-scheme: dark)')
    const dark = ref(media.matches)

    const update = () => dark.value = media.matches

    media.addEventListener('change', update)

    onUnmounted(() => {
      media.removeEventListener('change', update)
    })

    return {
      dark,
      toggleDark() {
        dark.value = !dark.value
      }
    }
  }
}
</script>
```

```
<script>
export default {
  data() {
    return {
      dark: false,
      media: window.matchMedia('(prefers-color-scheme: dark)'),
      setting: localStorage.getItem('setting-dark') || 'auto'
    }
  },
  methods: {
    toggleDark() {
      this.setting = this.setting === 'dark' ? 'light' : 'dark'
    },
    update() {
      if (this.setting === 'auto')
        this.dark = this.media.matches
      else
        this.dark = this.setting === 'dark'
    }
  },
  watch: {
    setting(newValue) {
      localStorage.setItem('setting-dark', newValue)
      this.update()
    }
  },
  created() {
    this.media.addEventListener('change', this.update)
    this.update()
  },
  destroyed() {
    this.media.removeEventListener('change', this.update)
  }
}
</script>
```

```
<script>
import { onUnmounted, ref, watch } from 'vue'

export default {
  setup() {
    const media = window.matchMedia('(prefers-color-scheme: dark)')
    const dark = ref(media.matches)
    const setting = ref(localStorage.getItem('setting-dark') || 'auto')

    const update = () => {
      if (setting.v === 'auto')
        dark.value = media.matches
      else
        dark.value = (setting.value === 'dark')
    }

    watch(setting, () => {
      localStorage.setItem('setting-dark', setting.value)
      this.update()
    })

    media.addEventListener('change', update)

    onUnmounted(() => {
      media.removeEventListener('change', update)
    })

    return {
      dark,
      toggleDark() {
        setting.value = setting.value === 'dark' ? 'light' : 'dark'
      }
    }
  }
}
</script>
```

```
<script>
export default {
  data() {
    return {
      dark: false,
      media: window.matchMedia('(prefers-color-scheme: dark)'),
      setting: localStorage.getItem('setting-dark') || 'auto'
    }
  },
  methods: {
    toggleDark() {
      this.setting = this.setting === 'dark' ? 'light' : 'dark'
    },
    update() {
      if (this.setting === 'auto')
        this.dark = this.media.matches
      else
        this.dark = this.setting === 'dark
    }
  },
  watch: {
    setting(newValue) {
      localStorage.setItem('setting-dark', newValue)
      this.update()
    }
  },
  created() {
    this.media.addEventListener('change', this.update)
    this.update()
  },
  destroyed() {
    this.media.removeEventListener('change', this.update)
  }
}
</script>
```

Bad Smell !

```
<script>
import { onUnmounted, ref, watch } from 'vue'

export default {
  setup() {
    const media = window.matchMedia('(prefers-color-scheme: dark)')
    const dark = ref(media.matches)
    const setting = ref(localStorage.getItem('setting-dark') || 'auto')

    const update = () => {
      if (setting.v === 'auto')
        dark.value = media.matches
      else
        dark.value = (setting.value === 'dark')
    }

                 ('setting-dark', setting.value)

    media.addEventListener('change', update)

    onUnmounted(() => {
      media.removeEventListener('change', update)
    })

    return {
      dark,
      toggleDark() {
        setting.value = setting.value === 'dark' ? 'light' : 'dark'
      }
    }
  }
}
</script>
```

逻辑复用?

Options API

```
<script>
export default {
  data() {
    return {
      dark: false,
      media: window.matchMedia('(prefers-color-scheme: dark)')
    }
  },
  methods: {
    toggleDark() {
      this.dark = !this.dark
    },
    update() {
      this.dark = this.media.matches
    }
  },
  created() {
    this.media.addEventListener('change', this.update)
    this.update()
  },
  destroyed() {
    this.media.removeEventListener('change', this.update)
  }
}
</script>
```

- 可以，但都不太理想
  - Mixin
  - Renderless Component
  - Vuex

Composition API

复制，粘贴。

```
<script>
import { useDark } from './utils'

export default {
  setup() {
    const { dark, toggleDark } = useDark()
    // other logic

    return {
      dark,
      toggleDark
    }
  }
}
</script>
```

进一步复用

专注解决一个问题

```
export function useDark() {
  const system = usePreferDark()
  const setting = useLocalStorage('setting-dark', 'auto')

  const dark = computed({
    get() {
      return setting.value === 'auto'
        ? system.value
        : setting.value === 'dark'
    },
    set(v) {
      if (v === system.value)
        setting.value = 'auto'
      else
        setting.value = v ? 'dark' : 'light'
    },
  })

  return dark
}
```

```
export function usePreferDark() {
  const media = window.matchMedia('(prefers-color-scheme: dark)')
  const dark = ref(media.matches)

  const update = () => dark.value = media.matches

  media.addEventListener('change', update)
  onUnmounted(() => {
    media.removeEventListener('change', update)
  })

  return dark
}
```

```
export function useLocalStorage(key, defaultValue) {
  const data = ref(localStorage.getItem(key) ?? defaultValue)

  watch(data, () => localStorage.setItem(key, data.value))

  return data
}
```

进一步复用

```
export function useDark() {
  const system = usePreferDark()
  const setting = useLocalStorage('setting-dark', 'auto')

  const dark = computed({
    get() {
      return setting.value === 'auto'
        ? system.value
        : setting.value === 'dark'
    },
    set(v) {
      if (v === system.value)
        setting.value = 'auto'
      else
        setting.value = v ? 'dark' : 'light'
    },
  })

  return dark
}
```

```
export function usePreferDark() {
  const media = window.matchMedia('(prefers-color-scheme: dark)')
  const dark = ref(media.matches)

  const update = () => dark.value = media.matches

  media.addEventListener('change', update)
  onUnmounted(() => {
    media.removeEventListener('change', update)
  })

  return dark
}
```

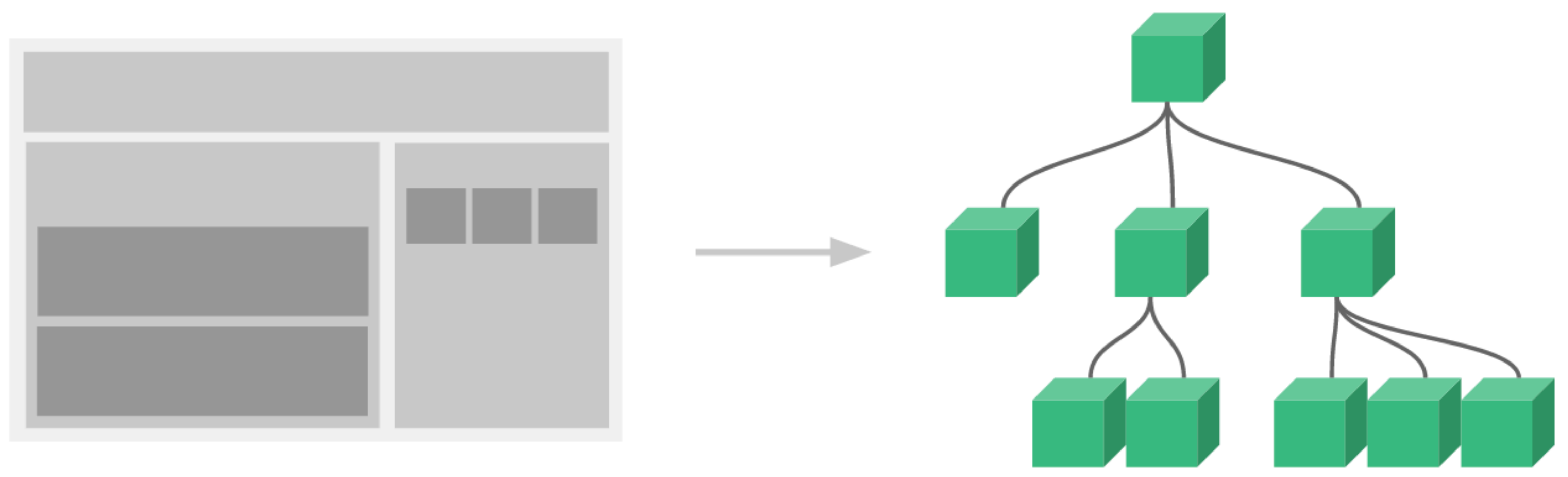
自动更新自动注销

```
export function useLocalStorage(key, defaultValue) {
  const data = ref(localStorage.getItem(key) ?? defaultValue)

  watch(data, () => localStorage.setItem(key, data.value))

  return data
}
```

自动保存

"逻辑组件"

Logical Components



UI 组件

Props → UI 🔄 State → 事件

逻辑组件

(响应式)参数 → (生命周期绑定) → 响应式数据 (ref, reactive, etc.)

现有的逻辑组件库
Composable Libraries

VueUse

细粒度的 Web API 与工具封装

- useEventListener
- useMouse
- useStorage
- useMediaQuery
- useDebounce
- useWebWorkerFn
- ...

vue-composable

常用逻辑的封装 by @pikax

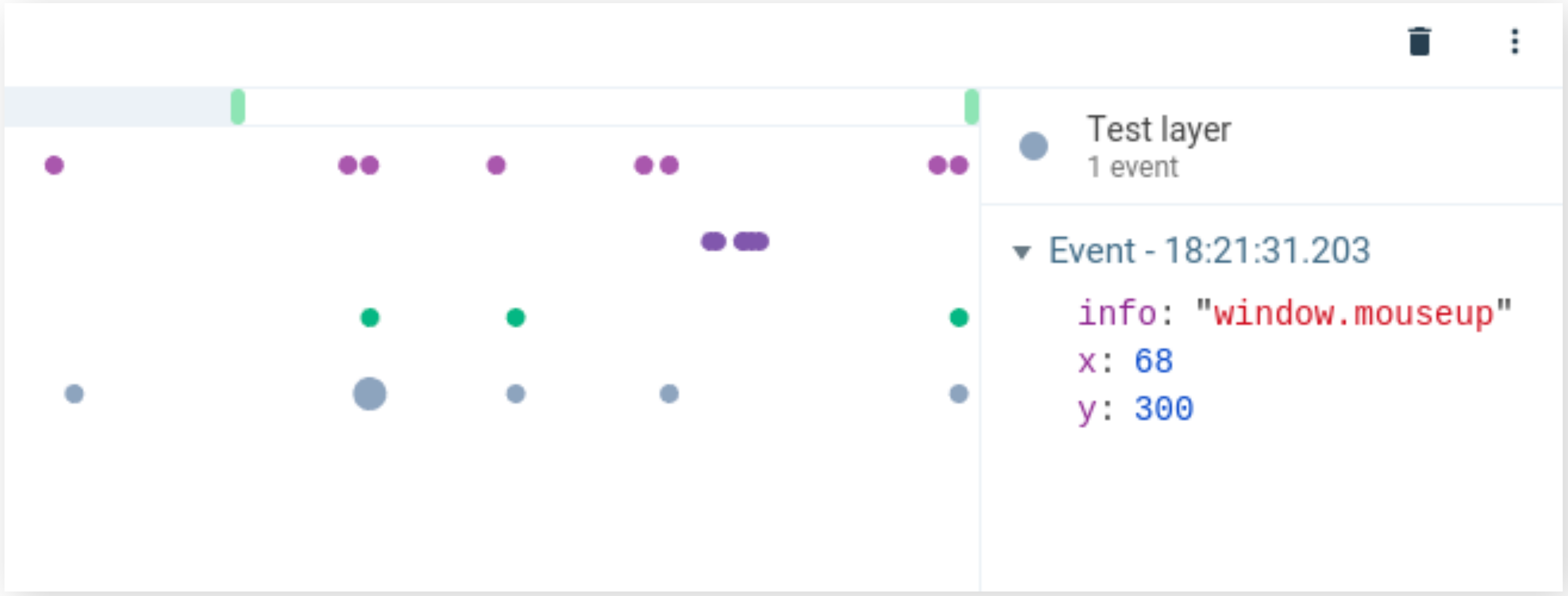
- useI18n
- useValidation
- useBreakpoints
- useDateFormat
- useUndo
- useSharedRef
- ...

组合式 API 生态支持

DevTools

6.0.0-beta-2 / @Akryum

- Vue 3 支持
- 自定义事件打点
- vue-composable by @pikax



Event - 18:21:31.203

```
info: "window.mouseup"
x: 68
y: 300
```

```
import { useDevtoolsInspector } from 'vue-composable'

const counter = ref(0)

useDevtoolsInspector({ counter })
```

SFC Improvements

<script setup>

```
<script>
import { ref, watchEffect, defineComponent } from 'vue'
import Foo from './Foo.vue'
import { useDark } from './utils'
import { store } from './store'

export default defineComponent({
  components: { Foo },
  setup() {
    const input = ref('')
    const dark = useDark()

    /* Other logics */
    watchEffect(() => console.log('input: ', input.value))

    return { store, input, isDark }
  }
})
</script>
```

```
<script setup>
import { ref, watchEffect } from 'vue'
import { useDark } from './utils'

export { default as Foo } from './Foo.vue'  注册组件
export { store } from './store'  导出全局状态到实例

export const input = ref('')
export const dark = useDark()  使用组合式函数并直接导出到实例

/* Other logics */
watchEffect(() => console.log('input: ', input.value))
</script>
```

Vue 2 & 3 代码同构

以工具库维护者的角度

Vue 2 & 3 同构

Isomorphic for Vue 2 & 3

```
// vue 3
import { ref, reactive } from 'vue'

// vue 2
import Vue from 'vue'
import plugin, { ref, reactive } from '@vue/composition-api'

Vue.use(plugin)
```

Vue 2 & 3 同构

Isomorphic for Vue 2 & 3

- 解决方案
    - 维护两个版本 (可以通过发布不同 tags 实现包名共享)
    - 编写一个发布脚本在打包的时候替换 API (一份代码，但还是要发布两个版本)

Vue Demi

Make Universal Library for Vue 2 & 3

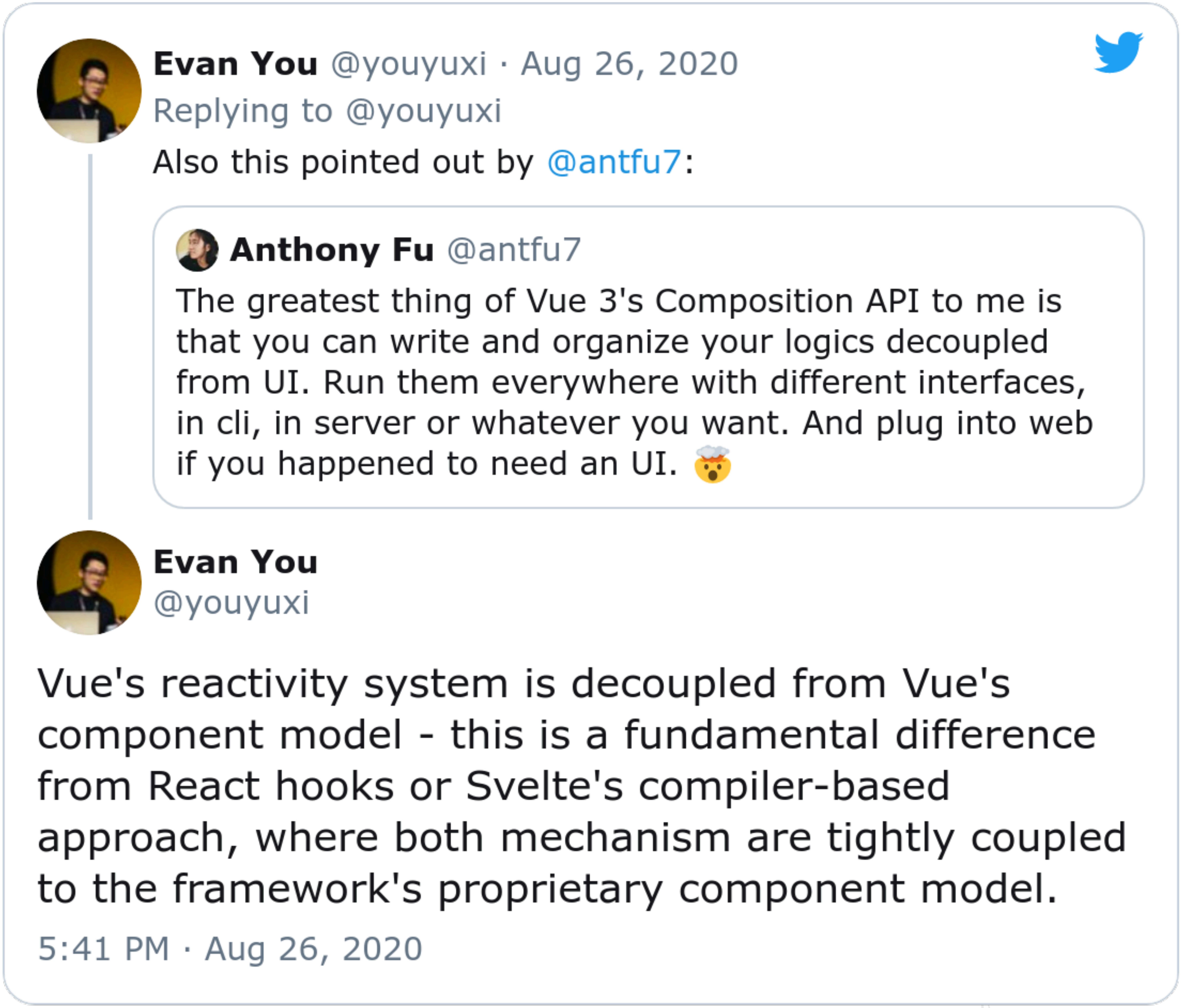
```
import { ref, reactive } from 'vue-demi'
```

- 只需要统一从 `vue-demi` 导入即可
- 通过 `postinstall` 检测当前 Vue 版本
- 根据用户环境 redirect 到对应的包
- 实现一个 npm 包同时支持 Vue 2 和 3
- 单元测试同构 by @pikax

- 当前使用 `vue-demi` 的库
  - `VueUse`
  - `@vue/apollo-composable`
  - `vuelidate`
  - `vue-use-infinite-scroll`

@vue/reactivity

灵活的响应式系统



Evan You @youyuxi · Aug 26, 2020
Replying to @youyuxi
Also this pointed out by @antfu7:

> **Anthony Fu** @antfu7
> The greatest thing of Vue 3's Composition API to me is that you can write and organize your logics decoupled from UI. Run them everywhere with different interfaces, in cli, in server or whatever you want. And plug into web if you happened to need an UI. 🤯

Evan You
@youyuxi

Vue's reactivity system is decoupled from Vue's component model - this is a fundamental difference from React hooks or Svelte's compiler-based approach, where both mechanism are tightly coupled to the framework's proprietary component model.

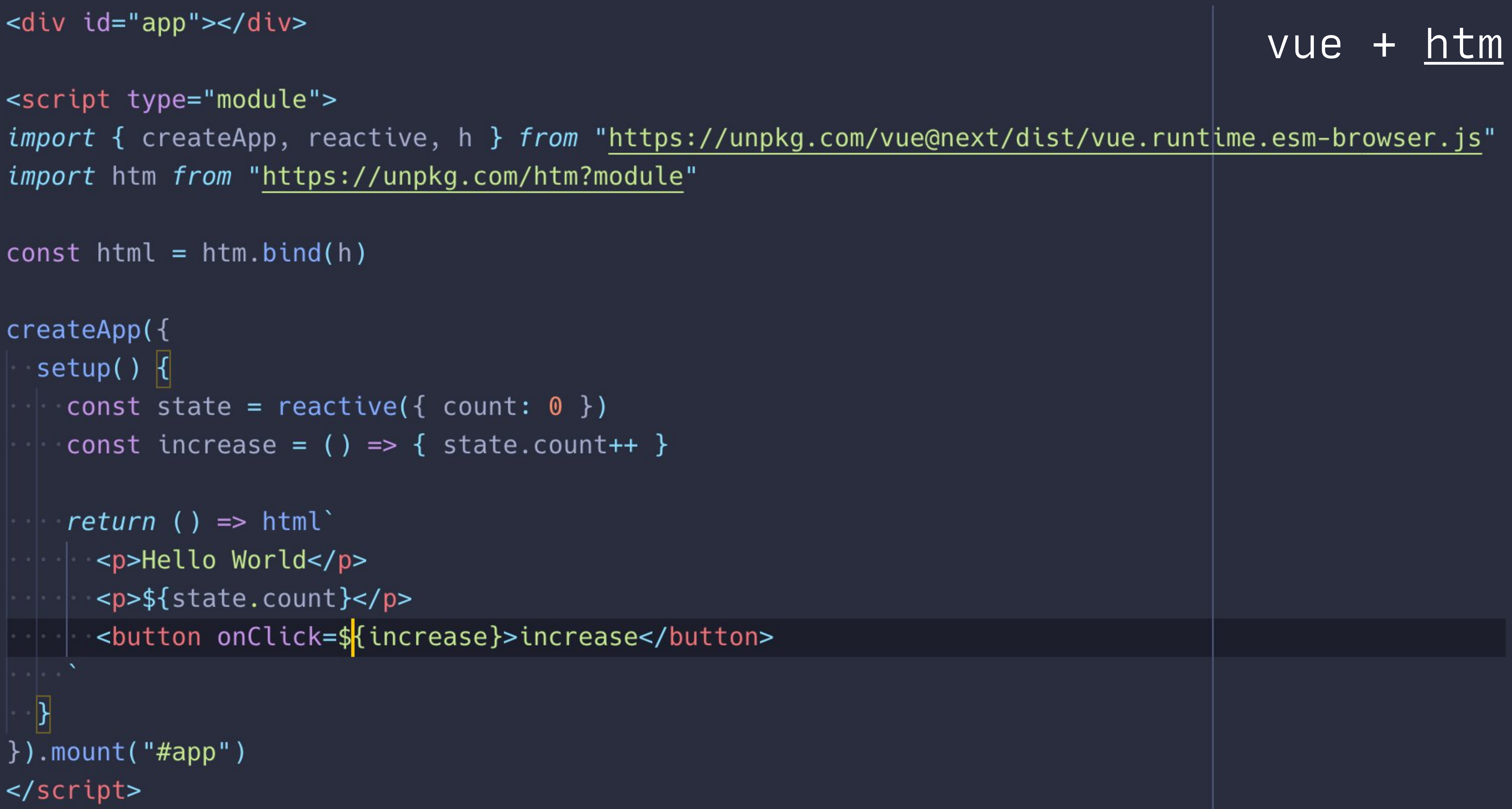
5:41 PM · Aug 26, 2020

UI 解耦

Decoupled from UI

- 响应式系统和 Vue 组件模型解耦
- 在不同环境/不同框架复用

vue + htm



```
<div id="app"></div>

<script type="module">
import { createApp, reactive, h } from "https://unpkg.com/vue@next/dist/vue.runtime.esm-browser.js"
import htm from "https://unpkg.com/htm?module"

const html = htm.bind(h)

createApp({
  setup() {
    const state = reactive({ count: 0 })
    const increase = () => { state.count++ }

    return () => html`
      <p>Hello World</p>
      <p>${state.count}</p>
      <button onClick=${increase}>increase</button>
    `
  }
}).mount("#app")
</script>
```

vue/reactivity + lit-html

```html
<div id="app"></div>

<script type="module">
import { html, render } from 'https://unpkg.com/lit-html?module'
import { reactive, effect } from "https://unpkg.com/@vue/reactivity/dist/reactivity.esm-browser.js"

const Component = () => {
  const state = reactive({ count: 0 })
  const increase = () => { state.count++ }

  return () => html`
    <p>Hello World</p>
    <p>${state.count}</p>
    <button @click=${increase}>increase</button>
  `
}

function mount(comp, target) {
  const template = comp()
  effect(() => render(template(), target))
}

mount(Component, document.querySelector('#app'))
</script>
```

reactivue (PoC)

在 React 中使用 Composition API

- 只依赖于 @vue/reactivity
- 包装了 React 的生命周期
  - onUpdated
  - onUnmounted
  - …
- 可复用 Vue 的工具库
  - VueUse
  - pinia

```
import React from 'React'
import { useSetup, ref, computed } from 'reactivue'

function MyCounter(props) {
  const { counter, doubled, inc } = useSetup(
    (props) => {
      const counter = ref(props.value)
      const doubled = computed(() => counter.value * 2)
      const inc = () => counter.value += 1

      return { counter, doubled, inc }
    },
    props
  )

  return (
    <div>
      <div>{counter} x 2 = {doubled}</div>
      <button onClick={inc}>Increase</button>
    </div>
  )
}

ReactDOM.render(<MyCounter value={10}>, el)
```

@vue-reactivity

关于 @vue/reactivity 可能性的探索

/**watch**
　实现 @vue/reactivity 中缺失的 watch

/**scope**
　Effect 自动收集

/**lifecycle** (wip)
　生命周期钩子

/**fs** (wip)
　响应式文件系统

/**bridge** (wip)
　客户端服务端响应式同步

…

```js
import { ref, reactive, computed } from '@vue/reactivity'
import { watch, watchEffect } from '@vue-reactivity/watch'

const count = ref(1)

const stopWatch = watch(
  count,
  (newValue) => {
    console.log(`Count: ${newValue}`)
  }
)

count.value += 1
// Count: 2

stopWatch()
```

@vue-reactivity

关于 @vue/reactivity 可能性的探索

/**watch**

实现 @vue/reactivity 中缺失的 watch

/**scope**

Effect 自动收集

/**lifecycle** (wip)

生命周期钩子

/**fs** (wip)

响应式文件系统

/**bridge** (wip)

客户端服务端响应式同步

...

```
import {
  effectScope,
  ref,
  computed,
  watch,
} from '@vue-reactivity/scope'

const counter = ref(0)

const stop = effectScope(() => {
  const doubled = computed(() => counter.value * 2)

  watch(doubled, () => console.log(double.value))

  watchEffect(() => console.log('Count: ', double.value))
})

// to dispose all effects
stop()
```

@vue-reactivity

关于 @vue/reactivity 可能性的探索

/**watch**

实现 @vue/reactivity 中缺失的 watch

/**scope**

Effect 自动收集

/**lifecycle** (wip)

生命周期钩子

/**fs** (wip)

响应式文件系统

/**bridge** (wip)

客户端服务端响应式同步

...

```js
import { effect } from '@vue/reactivity'
import { useJSON } from '@vue-reactivity/fs'

const data = useJSON('data.json')

// log on file changes
effect(() => {
  console.log(data.value)
})

// write back to file
data.value = { foo: 'bar' }
data.value.hello = 'world'
```

The Future?

References

- Vue.js 3.0 文档
- 官方仓库
  - Vue 3.0
  - @vue/composition-api (2.x 插件)
  - DevTools
- 第三方库
  - VueUse (逻辑工具库)
  - vue-demi (同构工具)
  - vue-composable (逻辑工具库)
  - swrv (SWR for Vue)
  - Vue Reactivity
- 实验性项目
  - reactivue

Thanks!

mail me at **hi@antfu.me**